# Packer

Easily build machines images for multiple platforms with the same configuration

http://packer.io

Lance Albertson - @ramereth - lance@osuosl.org

Oregon State University Open Source Lab

## About me

- Lance Albertson
- Director, OSU Open Source Lab (OSUOSL)
  - Provide infrastructure hosting for FOSS projects
  - Linux Foundation, Python Software Foundation, Drupal, etc
- Ops guy
- http://osuosl.org

## So what is Packer?

Machine image building tool created by Mitchell Hashimoto (of Vagrant fame)

Written in Go

Makes your life easier

## Supported Platforms

| | |
|---|---|
| Amazon EC2 | Digital Ocean |
| Docker | GCE |
| Openstack | Parallels |
| QEMU (kvm) | Virtual Box |
| VMWare | |

## What problem does Packer solve?

- One image building tool to rule them all
- Single configuration to create images across multiple platforms
    - Cloud? Vagrant? Docker? -- YES!
- Integrates into the cloud/devops model well

## Terminology

| | |
|---|---|
| Templates: | JSON files containing the build information |
| Builders: | Platform specific building configuration |
| Provisioners: | Tools that install software after the initial OS install |
| Post-processors: | Actions to happen after the image has been built |

## Packer Build Steps

*This varies depending on which builder you use. The following is an example for the QEMU builder*

1. Download ISO image
2. Create virtual machine
3. Boot virtual machine from the CD
4. Using VNC, type in commands in the installer to start an automated install via kickstart/preseed/etc
5. Packer automatically serves kickstart/preseed file with a built-in http server

## Packer Build Steps

6. Packer waits for ssh to become available
7. OS installer runs and then reboots
8. Packer connects via ssh to VM and runs provisioner (if set)
9. Packer Shuts down VM and then runs the post processor (if set)
10. PROFIT!

## Variables in Packer

Variables allow you to set API keys and other variable settings without changing the configuration file:

```
{
  "variables": {
    "aws_access_key": "",
    "aws_secret_key": ""
  },

  "builders": [{
    "type": "amazon-ebs",
    "access_key": "{{user `aws_access_key`}}",
    "secret_key": "{{user `aws_secret_key`}}",
  }]
}
```

## Environment Variables

You can also use variables to set environment variables within the packer environment that can be used by provisioners.

```
{
  "variables": {
    "my_secret": "{{env `MY_SECRET`}}",
  },
}
```

## Setting variables

You can set variables either via the CLI or importing them from a json file:

```
# Via CLI
$ packer build \
  -var 'aws_access_key=foo' \
  -var 'aws_secret_key=bar' \
  template.json

# Via json file
$ packer build -var-file=variables.json template.json
```

This makes it easy for you to adapt your automated builds as you need fit.

## How it works

Packer template file for QEMU:

```
{
  "builders": [
    {
      "boot_command": [
        "<tab> text ks=http://{{ .HTTPIP }}:{{ .HTTPPort }}/centos-7.0/ks-openstack.cfg",
        "<enter><wait>"
      ],
      "accelerator": "kvm",
      "boot_wait": "10s",
      "disk_size": 2048,
      "headless": true,
      "http_directory": "http",
      "iso_checksum": "df6dfdd25ebf443ca3375188d0b4b7f92f4153dc910b17bccc886bd54a7b7c86",
      "iso_checksum_type": "sha256",
      "iso_url": "{{user `mirror`}}/7.0.1406/isos/x86_64/CentOS-7.0-1406-x86_64-NetInstall.iso",
      "output_directory": "packer-centos-7.0-x86_64-openstack",
      "qemuargs": [ [ "-m", "1024m" ] ],
      "qemu_binary": "qemu-kvm",
      "shutdown_command": "echo 'centos'\|sudo -S /sbin/halt -h -p",
      "ssh_password": "centos",
      "ssh_port": 22,
      "ssh_username": "centos",
      "ssh_wait_timeout": "10000s",
      "type": "qemu",
      "vm_name": "packer-centos-7.0-x86_64"
  }],
```

# How it works

## Continued...

```
{
  "provisioners": [
    {
      "environment_vars": [
        "CHEF_VERSION={{user `chef_version`}}"
      ],
      "execute_command": "echo 'centos' | {{.Vars}} sudo -S -E bash '{{.Path}}'",
      "scripts": [
        "scripts/centos/osuosl.sh",
        "scripts/centos/fix-slow-dns.sh",
        "scripts/common/sshd.sh",
        "scripts/common/vmtools.sh",
        "scripts/common/chef.sh",
        "scripts/centos/openstack.sh",
        "scripts/centos/cleanup.sh",
        "scripts/common/minimize.sh"
      ],
      "type": "shell"
    }
  ],
  "variables": {
    "chef_version": "provisionerless",
    "mirror": "http://centos.osuosl.org"
  }
}
```

## Building the Image

```
$ packer build centos-7.0-x86_64-openstack.json
qemu output will be in this color.

==> qemu: Downloading or copying ISO
    qemu: Downloading or copying: http://centos.osuosl.org/7.0.1406/isos/x86_64/CentOS-7.0
==> qemu: Creating hard drive...
==> qemu: Starting HTTP server on port 8081
==> qemu: Found port for SSH: 3213.
==> qemu: Looking for available port between 5900 and 6000
==> qemu: Found available VNC port: 5947
==> qemu: Starting VM, booting from CD-ROM
    qemu: WARNING: The VM will be started in headless mode, as configured.
    qemu: In headless mode, errors during the boot sequence or OS setup
    qemu: won't be easily visible. Use at your own discretion.
==> qemu: Overriding defaults Qemu arguments with QemuArgs...
==> qemu: Waiting 10s for boot...
==> qemu: Connecting to VM via VNC
==> qemu: Typing the boot command over VNC...
==> qemu: Waiting for SSH to become available...
```

## Using the command line

```
# Build an image from a template
$ packer build template.json

# Inspect at template to see its configuration
$ packer inspect template.json
Optional variables and their defaults:

  chef_version = provisionerless
  mirror       = http://centos.osuosl.org

Builders:

  qemu

Provisioners:

  shell

# Validate proper json and packer configuration
$ packer validate template.json
Template validated successfully.
```

## Machine readable output

Most commands allow readable output for scripts:

```
$ packer inspect -machine-readable template.json
1424621191,,ui,say,Optional variables and their defaults:\n
1424621191,,template-variable,chef_version,provisionerless,0
1424621191,,ui,say,  chef_version = provisionerless
1424621191,,template-variable,mirror,http://centos.osuosl.org,0
1424621191,,ui,say,  mirror       = http://centos.osuosl.org
1424621191,,ui,say,
1424621191,,ui,say,Builders:\n
1424621191,,template-builder,qemu,qemu
1424621191,,ui,say,  qemu
1424621191,,ui,say,
1424621191,,ui,say,Provisioners:\n
1424621191,,template-provisioner,shell
1424621191,,ui,say,  shell
```

## Builders

- Responsible for creating and build the machines.
  - QEMU, Virtual Box, EC2, etc
- Builder definition maps to exactly one build
- You can have multiple builder definitions using the same builder
  - You must have a unique `name` for each build definition

| | |
|---|---|
| Amazon EC2 | Digital Ocean |
| Docker | GCE |
| Openstack | Parallels |
| QEMU (kvm) | Virtual Box |
| VMWare | |

## Amazon AMI Builder

amazon-ebs
: Create EBS-backed AMIs by launching a source AMI and re-packaging it into a new AMI after provisioning.

amazon-instance
: Create instance-store AMIs by launching and provisioning a source instance, then rebundling it and uploading it to S3

amazon-chroot
: Create EBS-backed AMIs from an existing EC2 instance by mounting the root device and using a Chroot environment to provision that device.

## Docker Builder

Builds docker images without the use of a `Dockerfile`

Able to provision containers with portable scripts that aren't tied to Docker itself

Allows you to use tools such as Chef, Ansible, etc to build the container

Must be run on a machine that already has docker installed

## Provisioners

| | |
|---|---|
| Shell | Run either inline or shell scripts |
| File Uploads | Upload files and use shell scripts to move files around as needed |
| Ansible | Provision using playbook and role files |
| Chef Client | Connect to a chef server and run chef |
| Chef Solo | Run a Chef solo run by pointing to local cookbooks or uploading them |
| Puppet Masterless | Run local manifests and modules |
| Puppet Server | Connect to a puppet server and run puppet |
| Salt | Using Salt states, deploy a vm using Salt |

## Post-processors

| | |
|---|---|
| compress | Compress VMWare or Virtualbox image using gzip |
| docker-import | Imports the docker image locally |
| docker-push | Push image to the docker repository |
| docker-save | Saves docker image directly to a file |
| docker-tag | Tags a build in the docker repository |
| Vagrant | Converts artifact into a valid Vagrant box file |
| Vagrant Cloud | Pushes artifact to Vagrant Cloud |
| vSphere | Uploads artifact to a vSphere endpoint |

## Extending Packer

You can extend packer using its plugin system

All builders, provisioners and post-processors are plugins themselves

Check out their documentation: https://packer.io/docs/extend/developing-plugins.html

## Other useful Packer tools

## Bento

https://github.com/chef/bento

- Chef's Packer template and script repository for building their vagrant boxes
- Covers most platforms you care about
- Figured out the hard stuff for you!
- Great place to see how to see Packer examples
- Checkout our fork: https://github.com/osuosl/bento/
  - QEMU Openstack builders for Ubuntu/Debian, CentOS/Fedora

## Demo time!

## Questions?

http://packer.io

Lance Albertson - @ramereth - lance@osuosl.org

Oregon State University Open Source Lab

http://osuosl.org